Title: Resource Allocation Strategies for Multiple Projects, Single Resource in a New Product Development Network

Course:
Year: 1991
Author(s): M. O. K. Gladhill, R. Phillips

Report No: P91014

Abstract: In this report we gain better understanding of some of the tools used in allocating resources for multiple simultaneous projects. Resource constrained multiple project resource allocation is most often considered using heuristics because optimal solutions quickly get to a size and complexity requiring tremendous computing power. Five heuristics were chosen for study based on previous research. A generic new product development project network, based on input from Precision Castparts Corporation titanium castings and Tektronix ASIC semiconductors, was used for the project network.

RESOURCE ALLOCATION STRATEGIES
for
Multiple Projects, Single Resource
in a
New Product Development Network

K. Gladhill, M. Olson, R. Phillips

EMP-P9114

Portland State University

Engineering Management Program

EMGT 510P

RESOURCE ALLOCATION STRATEGIES for Multiple Projects, Single Resource in a New Product Development Network

PROJECT TEAM:
Kristie Gladhil
Molly Olson
Robyn Phillips

June 4, 1991

Left out

Appendixes

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Resource constrained multiple project resource allocation is most often considered using heuristics because optimal solutions quickly get to a size and complexity requiring tremendous computing power. Five heuristics were chosen for study based on previous research. A generic new product development project network, based on input from Precision Castparts Corporation titanium castings and Tektronix ASIC semiconductors, was used for the project network. Four identical projects were started simultaneously, with a single resource for allocation. Trials were done for 100%, 50%, and 25% maximum resource allocation to any task. Figure 1 summarizes the scope of the study. Results indicate that heuristic performance depends on the network characteristics, as illustrated by the different results in the Precision Castparts and Tektronix networks. MAXTWK and SASP results differ from the other heuristics tested in that they spread out the range of project lengths.

The best heuristic will always depend on the user's overall objective criteria. Since heuristic performance is network dependent, the project manager would need to be able to try several heuristics to compare results. However, applying the heuristics involves a lot of effort, and is not practical for work situations. Inclusion of resource allocation scheduling heuristics in commercial project management software is recommended to provide project managers with the ability to better plan their programs.

PURPOSE

Projects often compete for the resources needed to accomplish their activities. Managers are put in the position of having to decide how to allocate a limited number of resources. Implicitly or explicitly, a manager has a prioritizing system. This study addresses the effect of several resource allocation strategies on new product development projects. The purpose of this project is to gain a better understanding of some of the tools used in allocating resources for multiple simultaneous projects.

BACKGROUND

Effective resource allocation is an important component of successful new product development. Organizations involved in design and manufacture in rapidly changing technologies often have many new product development projects occurring simultaneously. Reducing the time to market is becoming a common goal as the life cycle of these products becomes shorter.

A new product development project is composed of a series of tasks whose durations cannot be determined precisely. There is also interdependence of tasks, trying to schedule tasks in parallel, whenever possible, to complete projects more quickly. An additional complication is disagreement over how many simultaneous tasks a single engineer can be handle effectively. These three elements: high uncertainty, high interdependence of activities, and simultaneous task assignment; make effective

resource allocation even more difficult. The need for short product development cycles means that these decisions must be made quickly and accurately.

A project is a collection of activities that uses specific resources [1]. The term constrained project schedule refers to situations where assignment of a start time for a number of activities may be delayed because there are limited resources available [2]. Resource constrained project scheduling is an approach for minimizing project duration when there are fixed resources [3].

Resource limitations require sequencing decisions to resolve the resource scheduling conflicts [4]. There has been a lot of work on the subjects of resource allocation, project scheduling, and project compression; but several unrealistic simplifying assumptions are often made. Much of the literature has assumed unlimited resources. In reality this is rarely the case. Most of the existing research has been for single projects; there has been a limited amount published on simultaneous scheduling of multiple projects.

A factor which has been examined in earlier research is the parallel versus serial approach. In the parallel approach activity priority is determined as the schedule progresses, rather than in advance, as in the serial approach. Past research has indicated that parallel routines are superior to serial ones

[5]. Therefore, much of the recent research has focused on the parallel approach.

Research on project scheduling has been split into two approaches: optimization and heuristic procedures. The optimizing techniques, usually involving mathematical programming or analysis, have for the most part dealt with single projects rather than a multiple project situation. These techniques have been divided between integer linear programming or enumerative techniques. Optimizing techniques become quite cumbersome and require large amounts of computer capabilities as the size of the project increases. In general, there has been a lack of success implementing optimization procedures in common practice because of the limitations in the size of the problems a computer can handle [6]. As a result, there has been increased emphasis on developing heuristic procedures.

A project scheduling heuristic is defined by Davis and Patterson as a rule used to assign priorities based on a specific characteristic [7]. The heuristic rule is used to assign a priority when making scheduling decisions to resolve resource conflicts [8]. The solution obtained using heuristics is not necessarily optimal. However, it should give a reasonably good solution for a feasible amount of effort.

Research has shown two categories of heuristics to be most effective in minimizing the project duration [9]. The first

category uses a measure of time. Examples of this type of heuristic include job slack, job duration, and start/finish time. There are two general approaches to project time analysis: CPM (critical path methods) [10] and PERT (Program Evaluation Review Technique) [11]. CPM deals with resources in terms of resource levelling and considering "crash" costs; PERT does not address resources at all.

The second heuristic category incorporates a measure of resource usage [12]. One of the easiest methods to determine a project's resource requirements is to plot resource needs versus time [13]. After generating the resource requirement profile, it is compared against resources available. The resulting resource profile is sensitive to activity delays because all activities scheduled to start at the latest time are now critical [14]. This is a danger of resource levelling.

Generally, the resource allocation problem can be classified as time/cost problems, fluctuating resource demands, and project scheduling under fixed resource limits. Time/cost problems involve "crashing", getting additional resources to speed up completing tasks with the added expenses associated [15]. Evaluating the different possible combinations of task expedite costs requires special mathematical procedures. Resource levelling addresses fluctuating resource demands by rearranging the job scheduling within the available slack limits to achieve a better resource allocation distribution [16]. When resources are

fixed, leveling may not be able to solve the problem without having the finish date slip. Scheduling with fixed resources differs from resource levelling in that project duration is not allowed to increase [17].

Unfortunately, there has been limited testing and comparison of the different heuristic sequencing rules. Comparison has been further complicated by the lack of commonly accepted names for the heuristics. As an example, the rule "select shortest duration task effort" showed up in several studies and had a different name in each: SIO, SA, and SOF [18,19,20]. Research has shown the effectiveness of specific heuristics varies with the specific problem [21]. Existing research on the best heuristics for assigning priority to resolve resource constraints has been inconclusive. There have been conflicting results as to the best heuristic, even using the same objective function for evaluation. Unfortunately, little is known about the characteristics of the projects that the tests were run on. Therefore, it has not been possible to try to explain the different results. It has been concluded that it is not possible to tell in advance which heuristic will be the most effective for a specific scheduling problem [22].

Davis and Patterson compared the success of different heuristics using a parallel approach, with the purpose of identifying the best heuristic. The heuristics tested were chosen because they had been found effective in eleven earlier studies, only two of

which had compared the heuristic solutions to an optimal solution [23]. The heuristics used in the Davis and Patterson study are listed below [24]:

1.1 MINSLK or minimum job slack. Priority is given to the activity with the minimum slack, which is the difference between the critical path analysis Late Start Time (LST) and early start time (EST).

1.2 RSM, resource scheduling method, gives priority to the activity with the minimum value of d_{ij} where d_{ij} is the increase in project duration resulting when activity j follows activity i. $d_{ij} = \text{Max}\{0;(EFT_i - LST_j)\}$ where EFT_i = early finish time of activity i and LST_j = late start time of activity j. Comparison is made on a pair-wise basis among all the activities competing for resources at a specific point in time.

1.3 MINLFT, minimum late finish time, is the latest finish date allowed by the critical path that will not delay the project.

1.4 GRD, greatest resource demand rule bases priority on total resource unit requirements. It gives a higher priority to the activity with the greater resource demand. Priority = $d_j * \sum_{i=1}^{m} r_{ij}$ where d_j is the duration of the activity, r_{ij} is the per period requirement for resource type i by activity j and m is the number of different resource types.

1.5 GRU, greatest resource utilization, gives priority to the combination of activities that results in maximum resource utilization in each scheduling interval.

1.6 SIO, shortest imminent operation, is the shortest job first rule, based on the activity duration.

1.7 MJP, most jobs possible, assigns priority to the combination of activities that results in the greatest number of activities being scheduled in a given time period.

1.8 RAN, select jobs randomly, RAN.

Davis and Patterson ran eighty-three test problems and compared the result of each against the optimal solution. They found that MINSLK was the best heuristic in the majority of the trials [25].

This heuristic had previously been found to be the most effective by Mize, Fendly, and Patterson in a multi-project scheduling problem [26]. The next best heuristics were RSM and LFT. Davis and Patterson concluded that network structure, resource requirements, and resource availability may affect which heuristic is really the most effective. Thus the best heuristic is dependent upon the characteristics of a specific project problem [27].

Kurtulus and E. W. Davis also presented a study which evaluated the performance of different heuristics for multiple project scheduling rules. In their study, they assumed no pre-emption: once an activity was started its progress was not interrupted. [28]. In addition, they assumed the amount of resource required by a specific activity was constant, as was the amount of a resource available per period. Kurtulus and Davis studied the three heuristics most favored in previous research as well as six new heuristics which they proposed. The purpose was to compare the performance of the nine total heuristics in achieving the objective of minimizing the total project delay.

The nine sequencing rules are listed and defined below [29]:

2.1 SOF - Shortest Operation First, defined above in 1.6.

2.2 MINSLK - minimum slack first, defined above in 1.1.

2.3 SASP - the shortest activity from shortest project. Priority is given to the Minimum f_{ij} where $f_{ij} = CP_i + d_{ij}$. CP_i is the duration of the critical path of the project; d_{ij} is as previously defined in 1.2 above.

2.4 LALP is the longest activity from the longest project. LALP = Maximum f_{ij} where $f_{ij} = CP_i + d_{ij}$. CP_i and d_{ij} are the same as previously defined.

2.5 MOF is the maximum operation first and equals Max d_{ij}. This heuristic does not consider project duration as LALP does.

2.6 MAXSLK equals the maximum slack first.

2.7 MINTWK equals minimum total work content. MINTWK = Min f_{ij}. Let r_{ijk} represent the resources required by the activity. It then follows that $f_{ij} = TWK_i + d_{ij} * \sum_k r_{ijk}$ and $TWK_i = \sum_j \sum_k d_{ij} r_{ijk}$ for previously scheduled tasks. These define the total work content of the activities already scheduled from project i.

2.8 MAXTWK is the maximum total work content and equals Maximum f_{ij} where f_{ij} is defined the same as it was in the MINTWK definition, 2.7 above.

2.9 FCFS is the first come first serve heuristic. It means that the first eligible activity is assigned the highest priority to the resource.

SOF, MINSLK, and FCFS were the established measures that previous research had judged superior, although the order of performance was not conclusive. Kurtulus and Davis concluded from their study that two of their new heuristics, SASP and MAXTWK, outperformed the three heuristics previously thought to be superior [30].

Kurtulus and Davis [31] also found that the heuristics which were proven in earlier research to perform best in single project scheduling problems, SOF and LALP, were poor performers in the multiple project problem studied.

A final item pointed out in this paper was that many authors created a "single project" artificially from a multiple project problem, assuming that there would be no effect on resource allocation. However, this method was found to produce inferior results when compared to using multiple projects [32].

A study by Boctor [33] differed from previous approaches. He felt that, regardless of the characteristics of a specific problem, pre-selected combinations of heuristics could be used effectively. Boctor studied sixty-six problems using thirteen selected heuristics; five of these were for a serial approach. The eight heuristics tested in a parallel approach were [34]:

3.1 MINSLK - as defined above in 1.1

3.2 RSM - resource scheduling method as previously defined.

3.3 MINLFT - precedence to the activity having the smallest late finish time, the same as 1.3 above.

3.4 RAN - select an activity randomly

3.5 GRD - greatest number of resource units, the same as item 1.4 previously discussed.

3.6 SA - shortest activity; again, this heuristic is the same as 1.6 and 2.1 discussed in the previous studies.

3.7 SRD - smallest resource demand, the inverse of GRD.

3.8 LA - longest activity; the same as 2.5 above.

Boctor found that MINSLK, MINLFT, and RSM were the most efficient rules [35] with MINSLK best overall [36]. Boctor went on to discuss the best combinations of heuristics to use. He suggested the best strategy was to first try one of the most efficient heuristics and if necessary combine with the second most efficient one and so on. [37].

SCOPE

Based on previous work, we were interested in studying three heuristics from Davis and Patterson: MINSLK, RSM, and minLFT [38]; two from Kurtulus and Davis: SASP and MAXTWK [39]; and resource levelling in Timeline^{R}, a project management software package. These heuristics were chosen because they had shown the best performance in those previous studies.

Resource allocation for a single resource was studied for four identical projects starting simultaneously in each of the two network cases. Although this may not be realistic, it provided a consistent starting point for comparison of heuristics. Kurtulus and Davis also used this starting point for their study [40].

Resource allocation was considered for different maximum levels of resource allotment to any task: 100%, 50%, and 25%. The 25% level was expected to be more realistic since product engineers often have more than one project to work on at a time, and thus share time between several tasks. The 100% level would indicate concentrating on one task at a time until it was completed,

before another task could begin. The assumption of a constant amount of resource available was made in order to compare results with Kurtulus and Davis [41].

A "generic" project network for new product development was developed from experience in two very different industries: Precision Castparts Corporation's Titanium products, and Tektronix Integrated Circuits ASIC (Application Specific Integrated Circuit) semiconductor manufacturing. Two different cases of task effort and delays between tasks were then created for the network, one corresponding to each of the industries. The same predecessor/successor dependencies were used in both cases.

Heuristics were applied to determine the next starting task(s) using the parallel approach. The no pre-emption rule was utilized: once a task was started, it could not be interrupted. Figure 1 shows the trials studied.

EXPERIMENTAL

Development of the "generic" new product development project network plan was done by getting information based on experience in project management in two companies from the product engineering point of view. One of our team members is a product engineer on a development team at Precision Castparts and was able to provide information from her work making titanium

castings. The Tektronix information was obtained by interviewing a product engineer and manager from the work group of another team member. The network was based on their experience with new product development projects in ASIC semiconductor manufacturing. These plans were based on developing a new product for an established manufacturing process: a new part or design on an existing production process.

The new product development task sequences were found to be remarkably similar, even though the two companies have very different products and manufacturing processes (Figures 2 and 3). These development sequences were merged into a "generic" network (Figure 4). A brief description of each task is given below:

A. Received Order: This is a milestone task marking the beginning of the project.

B. Design: This task represents the design effort to meet the product specifications.

C. Process Route: The effort associated with developing the required manufacturing process routing through the factory.

D. Tool Acquisition: In both processes special tools that are purchased from outside suppliers are required for the manufacturing process. The effort associated with specifying and ordering this tool is included in this task.

D1. Delay associated with the manufacture of the tool is represented by a zero effort task of a duration appropriate for each company.

E. Test Plans: The effort required by the engineer to design a test procedure for the new product is included here.

F. Trial Run: The prototyping of the new product in both of these companies requires a trial manufacturing run. In both cases limited effort was required by the engineer over a fixed duration.

D2. Delay associated with preparation (by others) of the trial run product for testing by the engineer is represented here for Precision Castparts. For the Tektronix project, this represents delay for receiving testing probe cards from an outside supplier.

G. Test Trial Run: The effort associated with final test and evaluation as performed by the engineer is included here.

H. Customer Approval: The effort spent actively involved in reviewing and explaining the test results and product with the customer is represented by this task.

D3. Delay associated with customer review of the test results is included here.

I. Manufacturing Process Qualification: The effort spent by the engineer revising/reviewing the manufacturing process before it is sent to production is represented by this task for Precision Castparts. For the Tektronix project, this task is for additional prototype runs, again using limited effort for a fixed duration.

J. Documentation: Documents required for manufacturing and test of the product must be completed before the part can be released for production.

Note: Each task is identified by the above task letters on the Timeline® reports and spreadsheet printouts.

Once the "generic" network was developed, two different cases of task effort/duration times were established with times representative of the Precision Castparts and Tektronix ASIC businesses, respectively (Figure 4). Most tasks were defined as taking a specific amount of effort, but in both cases the "trial run", task F, only utilized a certain percentage of the available resource for a fixed time duration. For the Tektronix ASIC case, the manufacturing process qualification task also was for a fixed duration at a low resource usage level. Both network cases also included delays at certain points that are typical of waiting for the tool/masks to be received, or the probe card for the Tektronix ASIC test task.

The original plan was to use Timeline^{R} project management software, apply different heuristic plans to multiple projects on the same network case, and find the effect on project lengths and resource efficiency levels. However, it was found that Timeline^{R} did not have the ability to apply the heuristics automatically. Instead, the heuristics had to be applied "manually", stepping through the projects, task to task. The methods utilized for each heuristic are outlined in detail below. In each case, the primary heuristic was applied. If there were tie candidates for the next task(s) to be started, the decision was made using FCFS (first come, first serve), and if there still needed to be a distinction, a task from Project A was taken before Project B, B before C, and C before D, since the naming of the projects was essentially arbitrary.

Each heuristic was applied in parallel. The first set of tasks was allocated resources on the project start date. All other tasks were delayed until resources were again available. Task data (start date, end date, remaining project duration, and accumulated project effort) were recalculated using the end date of the completed tasks. The heuristic was then applied to the updated schedule to determine the next set of tasks to be allocated resources. This process was repeated until all tasks from all projects were complete.

Project length was an output from applying the heuristic. Resource efficiency was computed as amount of effort over time

span or the length of the longest of the four projects. Since the projects were identical, effort was the same for each trial of that network case. Therefore resource efficiency would be inversely proportional to the longest project length. Unlimited resource project duration was also calculated to use as a benchmark comparison for project length.

Timeline^{R} was not only incapable of using heuristics directly, but it also created some unrealistic restraints regarding utilization of resources. For example, during the trial run in the Tektronix network case, only 5% of the resource was being utilized. In practice a resource would most likely be used on other tasks with the remaining 95% of available time. However, if the task was set up in Timeline^{R} as requiring 100% of the resource, Timeline^{R} would not start the next task until the full 100% of the resource was available. In the Tektronix network case trials this was handled by assigning other tasks at less than the maximum utilization for the trial, e.g. at 95% while a task using 5% of the resources was underway. Also, once a resource allocation level was set for a task, it could not be changed. Using the above example, once the 5% task completed, the task underway using 95% would not pick up the additional 5% of resource then available.

Here: Explain what you did when you could & not use Timeline.

MINSLK RULE

A spreadsheet file computing ES (early start), EF (early finish), LS (late start), LF (late finish), and slack (LF-EF) times was set up with the four projects for the two different network cases. Starting at TIME = 0, the candidate tasks to be worked on were identified as those having ES times less than or equal to the TIME value. The spreadsheet was set up such that a time could be input, and all ES times on tasks not yet begun would be set to that time value to represent the situation at that time in the project cycle.

For MINSLK, the column giving slack was checked on candidate tasks, and the lowest value chosen. If more than one task tied for the lowest slack value, the FCFS rule was then applied. If there was still a tie, the preference was arbitrarily given to the first project listed. Appendix A gives a step by step printout of the spreadsheet through the project cycle.

RSM

The RSM heuristic was applied using the same spreadsheet file to update information with time for the project, and an additional spreadsheet which had a matrix to give the $d_{ij} = \text{Max}\{0, (EF_i - LS_j)\}$ values for the candidate tasks available for starting at time t. The lowest d_{ij} value was chosen, which would indicate the task from the LS value to be started. FCFS and project list order were the back up rules for ties, as used in MINSLK. Appendix B has an example of the RSM rule applied.

MINLFT

This heuristic was also applied utilizing the spreadsheet, with the LF times instead of MINSLK determining the next task to be chosen. FCFS and project list order were the secondary and tertiary rules to break ties. An example is given in Appendix C.

SASP

This heuristic prioritizes tasks by first selecting the project with the shortest remaining critical path length and then assigning resource to the available task with the shortest duration. This heuristic is "forward looking" in that it sees the remaining path duration. However, it should be noted that the duration it sees is the unleveled duration.

The SASP heuristic was implemented using Timeline^{R} project management software. Four identical projects were combined into one project schedule. Each project was given the same start date, in the future (July 1, 1991) to avoid Timeline^{R} trying to update tasks as we worked on the schedules. Delays were modeled as duration driven with zero effort required. Timeline^{R} allows a priority to be assigned to each task. All priorities were set to 1 (highest priority) initially and reset in order of resource allocation. Filters, layouts, and sorting were done, as described in Appendix D, to simplify usage. Final results were verified by reviewing resource allocation histograms and by comparing results to a Timeline^{R} levelled schedule using the priorities assigned during the implementation process.

MAXTWK

This heuristic selects the next task for resource allocation based on task effort. Effort is the duration of the task times the percent resource allocation required for the task. This characteristic remains constant with respect to the leveling process. If the available resource is less, the duration increases proportionately. This heuristic is "backward looking" in that it prioritizes the projects by most spent effort. The project with the most time and resource invested is assigned the highest priority. The available task with the most time and resource required within this project is then assigned the next available resource.

The same process as was described in the SASP write-up for using Timeline^{R} applies.

Timeline^{R} Resource Levelling

The networks were input into Timeline^{R}, with the partial dependencies used for delays and resource allocations specified. Resource levelling was applied, and Timeline^{R} chose tasks based on its rules. Timeline's^{R} priority system for resolving resource levelling are as follows:

1. Fixed date tasks have priority over ASAP (as soon as possible) or ALAP (as late as possible) tasks.
2. Earlier scheduled start dates have priority over later ones.
3. Longer duration tasks have priority over shorter ones.

DISCUSSION

Appendix E contains Timeline^{R} Gantt charts of the trials shown in Figure 1.

MINSLK discussion

MINSLK looks ahead to see which tasks have the least slack, and schedules them ahead of tasks with slack remaining. Many of the choices needed to go to the tie breakers in applying the heuristic to the cases studied. For the Precision network case at 25% resource allocation, there was time with no resource utilization due to task delays.

RSM discussion

The RSM heuristic was cumbersome and time consuming to apply even in this rather simplistic trial. For a large period of time the pair wise comparisons resulted in a tie among several activities. This was especially true during the 100% trial when each task took 100% of the resource. The FCFS heuristic was then applied and was the deciding factor for many of the required decisions.

For less than 100% resource allocation, even though more tasks could have been done at once, the time that the resource efficiency was below 100% increased from 110 days in the 50% trial to 160 during the 25% trial. The low resource efficiency was caused by the delays in the project itself. The delays ended up driving the project completion times.

MINLFT discussion

MINLFT looks ahead in the project plans to the task which will be completed the soonest. With the spreadsheet for critical path analysis, this was fairly easy to apply manually. The Precision trials had times with no resource utilization due to delay in the schedule.

SASP discussion

As many tasks in the network are on the critical path, completion of a task tends to shorten that project's critical path.
This heuristic tends to select a project and assign resources to it until it no longer requires all available resources because of the use of identical projects. This was, of course, more true of the higher resource allocation levels (50% and 100%) since these levels tended to strengthen the effect of the heuristic.
The lower level allocation (25%) tended to weaken the prioritization of the heuristic since many competing projects were assigned similar levels of resource simultaneously.

Because this network was dominated by long duration tasks with little or no effort, this heuristic tended to have less spread between project durations than the MAXTWK heuristic which emphasized effort over duration.

MAXTWK Discussion

This heuristic tended to stay with a project until that project could no longer consume full resources. At that time, it moved to assigning resources to the next project until the first project required resource again or until the second project had surpassed the first in consumed effort. A wide range in project completion times of simultaneously started projects was seen, which was consistent with the described behavior.

Because all four projects were identical, the heuristic gave highest priority to tasks from project A, then B, then C, and then D in the case of 100% resource level. At 50% resource level, the spreading became more pronounced as more of the parallel tasks at the beginning of project A were assigned resources prior to other projects starting. At 25% there was sufficient resource to provide for more tasks than Project A had available. The project spread was less since other projects began sooner.

The heuristic gave slightly better results for 50% allocation than it did for 100%. The performance degraded significantly at 25% resource level. This would tend to support a philosophy of assigning fewer tasks to a human resource if using this heuristic. More study is required however, since the network delay durations significantly affect this result. The Timeline^{R} requirement for constant task resource allocation once a task starts also alters the model's performance from reality.

Tools discussion

Another issue under study was the capability of readily available commercial project management software tools to implement the heuristics. Project management software was chosen, in lieu of special purpose computer code, because it is more likely to be in use when multiple projects are being administrated. The ease or difficulty of using resource allocation heuristics was believed to be an important factor in determining whether they would be implemented in practice. This is particularly true because of the variable performance of heuristics.

Most commercially available project management software packages resolve resource conflicts using levelling. Unfortunately, the built-in leveling heuristics typically are not very effective and the user is not allowed to specify alternate resource conflict resolution heuristics.

The user can try to override this system by assigning priorities to each task. The difficulty is in ascertaining these priorities. The process of manually applying the heuristic was time consuming and fraught with potential for error. Each of the four projects had thirteen tasks. At each iteration the user had to review fifty-two tasks to see which were available for starting, which were already in progress, and how much resource was available for allocation. Once the contenders were identified the heuristic was applied and the resource was allocated. The in-process task with the earliest end date then

had to be identified from the fifty-two tasks and the current date updated. If an error was made it was difficult to reconstruct the project schedule accurately. The process must be run "in reverse" in order to correct the problem.

Potential errors fell into several categories. Understanding the cases of the errors points to methods that will minimize them. In general, errors occurred when:

1) User selected incorrect task for resource allocation because of the large number of tasks to review.

2) User pushed the wrong key and did not perform the function desired.

Reduction of the potential for these errors could be accomplished by minimizing the data the user must review, displaying it to assist the user in choosing the correct task, and reducing the number of keys the user must press in order to perform choice, selection, start, and update functions. Timeline^{R} does allow the user to customize the display layout to show only information that is relevant to the decision at hand.

Evaluation Criteria

There are a variety of criteria by which one can evaluate the behavior of the heuristics.

If one assumes the value of the projects to the company to be similar, then the greatest throughput of projects in the shortest time would be a good measure of the value of the heuristic.

Given that projects often have different expected payback one could evaluate heuristics based on which completes the project with highest expected return as quickly as possible.

Another method for evaluating heuristic performance would be to observe resource usage efficiency over the course of the projects.

If it is desirable to complete projects in a consistent amount of time (all projects take the same amount of time) then one would look for a heuristic that minimizes the time difference between the first and last project to be completed.

Lastly, it is of interest to see how the different philosophies of resource level affect the heuristic's performance. If a company is considering concurrent engineering as a method for reducing time to market they may well wonder how the practice of dedicating a resource to one task at a time will affect duration.

RESULTS

The greatest throughput of projects was measured by the minimum average project length. The average project lengths are summarized in Table 1. The performance of the heuristics was strongly influenced by the resource allocation (Figures 5 and 6). Table 2 lists the heuristics for each network and resource level from highest to lowest throughput. MINSLK showed the most dramatic change, moving from best to worst as resource level

changed. SASP was least sensitive to resource level, giving good results in all three cases. The heuristics applied to the Tektronix network are summarized in Table 3. This network was extremely sensitive to resource level. MINLFT showed the least sensitivity. RSM, Timeline^{R}, and MINLFT throughput decreased for the 25% resource level. However, SASP, MAXTWK, and MINSLK all had increased project throughput at the 25% level.

Resource conflicts occurred less often as the resource allocation level decreased because more tasks were being allocated resource simultaneously. The 25% resource allocation trials thus reflect less influence from the primary heuristic. It was also apparent that the network delays and limited effort/fixed duration tasks had a strong effect on heuristic performance.

Resource efficiency was calculated as the effort for the project divided by the duration of the longest project. Since all four projects required identical effort for a network case, this was a measure of the longest or worst case project duration the heuristic generated. Tables 4 and 5 rank the heuristics from best to worst efficiency for the two networks cases. MINSLK and RSM provided the highest efficiencies across all resource levels. MAXTWK and SASP were consistently the lowest. Efficiency on the Tektronix network again showed sensitivity to resource level.

Figures 7 - 12 show the four project durations for all trials. It is interesting to note that the SASP and MAXTWK heuristics

yielded both the longest and shortest project lengths. Range of duration between simultaneously started projects was also greatest for those two heuristics. SASP and MAXTWK consider total project data as well as task data; all other heuristics considered individual tasks only. MINSLK, RSM, and MINLFT tended to generate much more consistent durations across all four projects.

The ratio of average project length to unconstrained resource project length is plotted in Figures 13 and 14. The unlimited resource length is simply the summation of the durations of the critical path activities. The purpose of the ratio was to provide a benchmark for comparison of heuristically levelled project length with unlimited resource project length. Both networks showed the same general trend with heuristic project lengths approaching unlimited resource lengths as percent resource level moves to 25%. It should be remembered that, at 25% resource allocation, the heuristics had less opportunity to influence the schedule.

CONCLUSIONS

* Heuristic performance depends on the network characteristics as illustrated by the different results in the Precision Castparts and Tektronix networks.

* In both networks, MAXTWK and SASP give substantially different results for multiple projects than do the other heuristics tested. SASP and MAXTWK spread out the range of project lengths.

* The best heuristic is a function of the user's overall objective criteria. Best throughput, fastest time to completion of an individual project, or shortest worst case length criteria result in different heuristics being chosen as best.

* Project management software needs to incorporate automated heuristic options for heuristics to be useful. Manual calculation of project duration using heuristics applied in parallel is too tedious and error prone. Since the heuristic behavior is network dependent, the project manager would need to be able to try several heuristics to compare results.

RECOMMENDATIONS FOR FURTHER WORK

* An optimal solution for the network cases would be useful as a benchmark. Criteria for optimization would have to be established. Two candidate criteria could be maximum throughput of projects and best return on investment by finishing highest return projects first.

* Multiple resource types could be considered, instead of the single resource.

* More networks could be studied, with different dependencies and delays. It would be interesting to characterize new product development networks for a particular industry.

* Simulation of the heuristic performance would be an excellent idea since the durations and efforts of the tasks in new product development are not deterministic as this study has assumed. A simulation program would also allow the resource assigned to a task to make use of all the available resource time instead of having to be a fixed percentage for the entire task duration.

* Different levels of maximum resource allocation, other than the 100%, 50%, and 25% levels could be studied in order to understand the effects on heuristic performance.

* Further work on measures to compare networks could be done to allow heuristic comparisons between dissimilar projects.

* Automation of these heuristics in commercially available software such as project management software, spreadsheets and database programs is required to make application practical. The time that is required to characterize the performance of the heuristic in a particular network is of such a magnitude to discourage their use in common practice.

FIGURE 1

SCOPE OF PROJECT WORK

heuristics	Precision Castparts network case			Tektronix ASIC semiconductor network case		
	100%	50%	25%	100%	50%	25%
MINSLK	X	X	X	X	X	X
RSM	X	X	X	X	X	X
MINLFT	X	X	X	X	X	X
SASP	X	X	X	X	X	X
MAXTWK	X	X	X	X	X	X
Timeline^{R}	X	X	X	X	X	X

Did you have one Seven project networks, or two different "cases"?